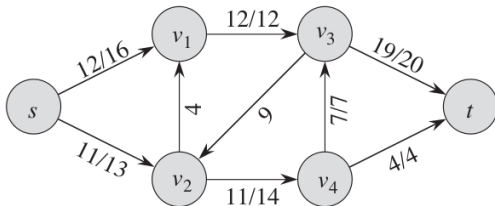


# Fluxo Máximo

Pedro Ribeiro

DCC/FCUP

2020/2021

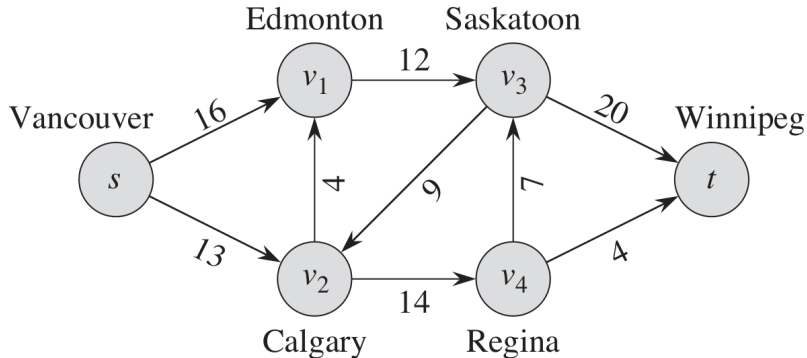


# Fluxo Máximo

- Um grafo pesado pode ser interpretado como uma **rede de canalizações** onde o peso é a **capacidade** de cada canalização.
- Calcular o **fluxo máximo** implica descobrir qual o fluxo máximo que pode ser enviado de um nó  $s$  (*source*) para um nó  $t$  (*sink*).
  - ▶ Imagine água a circular nos canos e o que quer é maximizar o volume de água a circular entre  $s$  e  $t$
- Mais formalmente, sendo  $f(u, v)$  o fluxo na aresta  $(u, v)$ , e  $c(u, v)$  a sua capacidade, temos de obedecer às seguintes **restrições**:
  - ▶ **Capacidades**:  $0 \leq f(u, v) \leq c(u, v)$  para qualquer  $(u, v)$   
(O fluxo não é negativo e tem de ser menor ou igual à capacidade)
  - ▶ **Conservação de Fluxo**: Para cada  $u \in V - \{s, t\}$ ,  
$$\sum_{v \in V} f(v, u) = \sum_{v \in V} f(u, v)$$
  
(Em cada nó que não a origem e o destino, a soma do fluxo que entra é igual à soma do fluxo que sai)
  - ▶ **Máximo Fluxo**: Queremos maximizar  $|f|$ , onde  
$$|f| = \sum_{v \in V} f(s, v) - \sum_{v \in V} f(v, s)$$
  
(Fluxo total é o que sai da origem menos o que entra na origem)

# Fluxo Máximo

- Vejamos um exemplo. Imagine a seguinte rede:

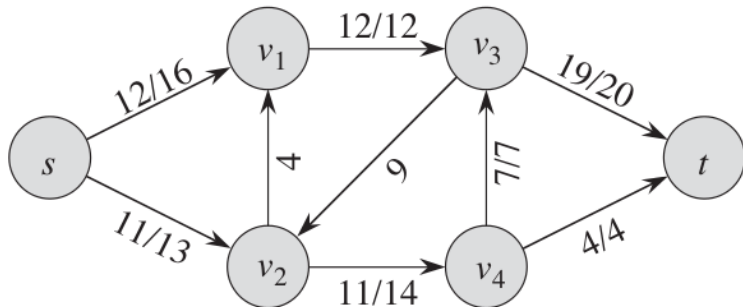


(imagem de Introduction to Algorithms, 3rd Edition)

- Qual o **fluxo máximo** entre *Vancouver* e *Winnipeg* ?

# Fluxo Máximo

- O fluxo máximo seria **23** e podia ser obtido da seguinte maneira:  
(a/b nas aresta indica fluxo/capacidade)



(imagem de Introduction to Algorithms, 3rd Edition)

Note que 23 é a soma do fluxo que sai da origem (12 através da aresta  $(s, v_1)$  e 11 através da aresta  $(s, v_2)$ )

# Algoritmos para Fluxo Máximo - Ford-Fulkerson

- Dada uma **rede de fluxos** como calcular o seu fluxo máximo?
- Uma hipótese é usar o **método de Ford-Fulkerson**  
(chamamos de método porque é uma ideia que pode ser concretizada depois com vários algoritmos)

## Método de Ford-Fulkerson: fluxo máximo no grafo $G$ , de $s$ para $t$

Ford-Fulkerson( $G, s, t$ ):

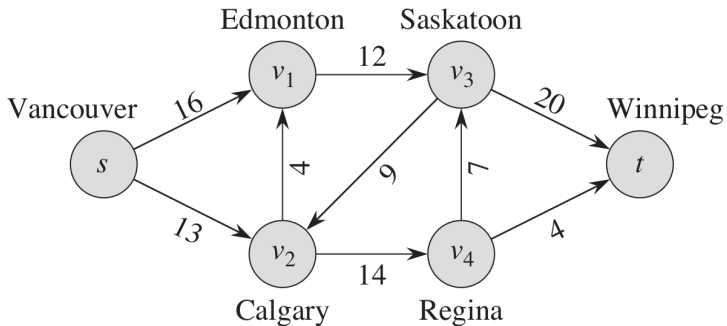
Inicializar fluxo  $f$  a zero

**Enquanto** existir um *caminho de aumento*  $p$  no grafo residual  $G_f$  **fazer**:  
    aumentar fluxo  $f$  ao longo de  $p$   
**retornar**  $f$

- Um **caminho de aumento** (*augmenting path*) é um caminho pelo qual ainda é possível enviar fluxo
- Um **grafo residual**  $G_f$  é um grafo que indica como podemos modificar o fluxo nas arestas de  $G$  depois de já aplicado o fluxo  $f$ .

# Ford-Fulkerson passo a passo

- Vejamos agora um exemplo do Ford-Fulkerson passo a passo para que possa compreender bem os conceitos usados.

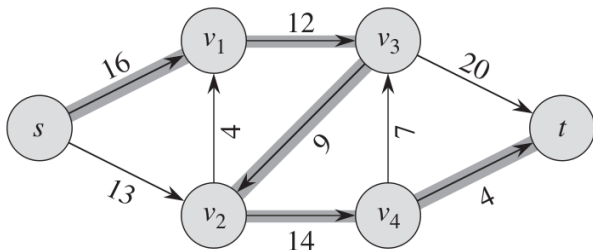


(imagem de Introduction to Algorithms, 3rd Edition)

- Este é o grafo inicial com as capacidades indicadas nas arestas.

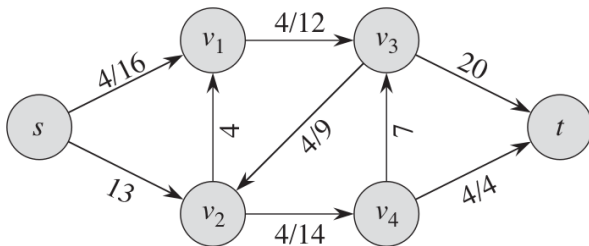
# Ford-Fulkerson passo a passo

- Um caminho de aumento é um caminho entre a origem  $s$  e o destino  $t$  que nos permite adicionar fluxo, ou seja, um caminho onde a capacidade mínima das arestas é maior que 0.
- No caso do nosso grafo existem vários caminhos de aumento. Entre eles está o caminho  $s \rightarrow v_1 \rightarrow v_3 \rightarrow v_2 \rightarrow v_4 \rightarrow t$ , indicado a cinzento.
- A capacidade mínima ao longo do caminho é 4 (mínimo entre 16, 12, 9, 14 e 4), pelo que podemos enviar um fluxo de 4.



# Ford-Fulkerson passo a passo

- Ao enviarmos o fluxo de 4 ao longo do caminho atrás indicado, os fluxos ficam do seguinte modo:  
(a/b nas aresta indica fluxo/capacidade)



(imagem de Introduction to Algorithms, 3rd Edition)

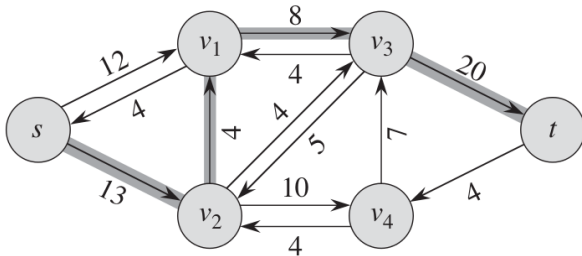


# Ford-Fulkerson passo a passo

- O grafo residual mostra onde podemos ainda aplicar fluxo.
- Depois de adicionarmos um fluxo  $f(a)$  longo de um caminho, o grafo residual é obtido fazendo as seguintes transformações ao longo de cada aresta  $(u, v)$  no caminho de aumento que escolhemos:
  - ▶ Na direção do caminho que tomamos, reduzimos o peso das arestas em  $f(a)$ , ou seja,  $c(u, v) = c(u, v) - f(a)$ . Se  $c(u, v)$  ficar a zero, retiramos a aresta. Isto representa a quantidade de fluxo que ainda podemos fazer passar pela aresta na direção original
  - ▶ Na direção oposta, aumentamos o peso da aresta em  $f(a)$ , ou seja,  $c(v, u) = c(v, u) + f(a)$ . Se a aresta não existia, cria-se. Isto representa que se quisermos podemos "retirar" fluxo ao longo desta aresta, o que pode dar jeito para aumentar depois via outro caminho.

# Ford-Fulkerson passo a passo

- Depois do fluxo de 4 indicado atrás, o grafo residual ficava como a imagem seguinte documenta.

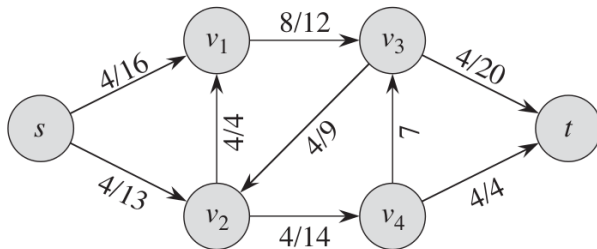


(imagem de Introduction to Algorithms, 3rd Edition)

- Este grafo residual ainda admite vários caminhos de aumento. Entre eles está o caminho  $s \rightarrow v_2 \rightarrow v_1 \rightarrow v_3 \rightarrow t$ . A capacidade mínima ao longo do caminho é 4 (mínimo entre 13, 4, 8 e 20).

# Ford-Fulkerson passo a passo

- Ao enviarmos o fluxo de 4 ao longo do novo caminho atrás indicado, os fluxos ficam do seguinte modo:  
(a/b nas aresta indica fluxo/capacidade)

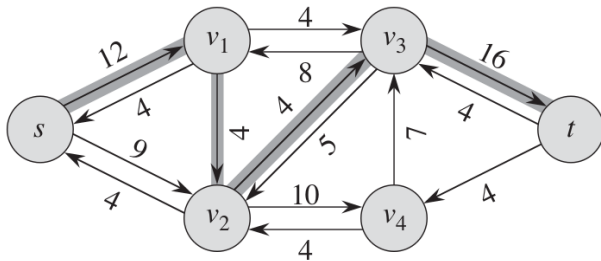


(imagem de Introduction to Algorithms, 3rd Edition)

- O fluxo total a sair da origem é agora de 8 ( $4 + 4$ ).

## Ford-Fulkerson passo a passo

- Depois do novo fluxo de 4 indicado atrás, o grafo residual ficava como a imagem seguinte documenta.

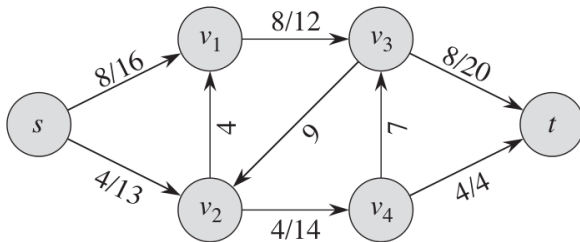


(imagem de Introduction to Algorithms, 3rd Edition)

- Este grafo residual ainda admite vários caminhos de aumento. Entre eles está o caminho  $s \rightarrow v_1 \rightarrow v_2 \rightarrow v_3 \rightarrow t$ . A capacidade mínima ao longo do caminho é 4 (mínimo entre 12, 4, 4 e 16). Note como está a ser usada a aresta  $(v_1, v_2)$  que tinha sido criada anteriormente.

# Ford-Fulkerson passo a passo

- Ao enviarmos o fluxo de 4 ao longo do novo caminho atrás indicado, os fluxos ficam do seguinte modo:  
(a/b nas aresta indica fluxo/capacidade)

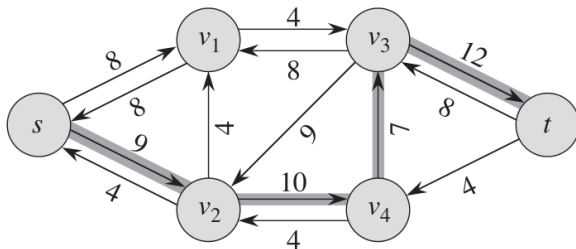


(imagem de Introduction to Algorithms, 3rd Edition)

- O fluxo total a sair da origem é agora de 12 ( $8 + 4$ ).

# Ford-Fulkerson passo a passo

- Depois do novo fluxo de 4 indicado atrás, o grafo residual ficava como a imagem seguinte documenta.

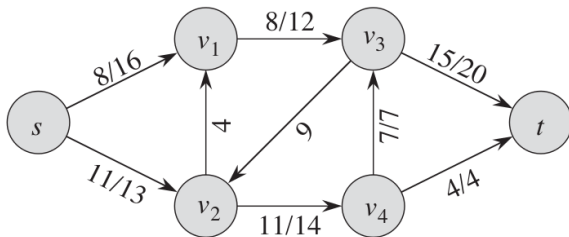


(imagem de Introduction to Algorithms, 3rd Edition)

- Este grafo residual ainda admite vários caminhos de aumento. Entre eles está o caminho  $s \rightarrow v_2 \rightarrow v_4 \rightarrow v_3 \rightarrow t$ . A capacidade mínima ao longo do caminho é 7 (mínimo entre 9, 10, 7 e 12).

# Ford-Fulkerson passo a passo

- Ao enviarmos o fluxo de 7 ao longo do novo caminho atrás indicado, os fluxos ficam do seguinte modo:  
(a/b nas aresta indica fluxo/capacidade)

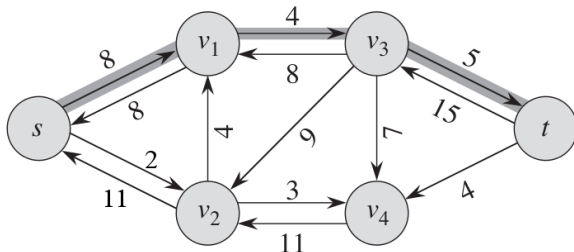


(imagem de Introduction to Algorithms, 3rd Edition)

- O fluxo total a sair da origem é agora de 19 ( $8 + 11$ ).

# Ford-Fulkerson passo a passo

- Depois do novo fluxo de 7 indicado atrás, o grafo residual ficava como a imagem seguinte documenta.



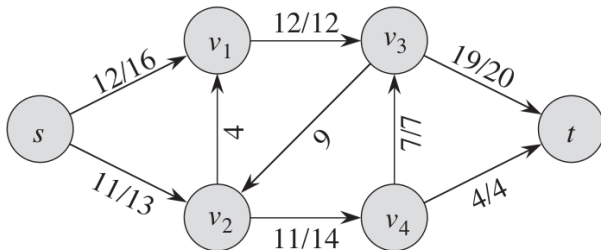
(imagem de Introduction to Algorithms, 3rd Edition)

- Este grafo residual ainda um caminhos de aumento:  
 $s \rightarrow v_1 \rightarrow v_3 \rightarrow t$ . A capacidade mínima ao longo do caminho é 4 (mínimo entre 8, 4 e 5).



# Ford-Fulkerson passo a passo

- Ao enviarmos o fluxo de 4 ao longo do novo caminho atrás indicado, os fluxos ficam do seguinte modo:  
(a/b nas aresta indica fluxo/capacidade)

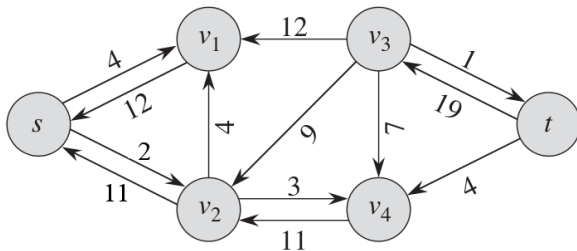


(imagem de Introduction to Algorithms, 3rd Edition)

- O fluxo total a sair da origem é agora de 23 (12 + 11).

# Ford-Fulkerson passo a passo

- Depois do novo fluxo de 4 indicado atrás, o grafo residual ficava como a imagem seguinte documenta.



(imagem de Introduction to Algorithms, 3rd Edition)

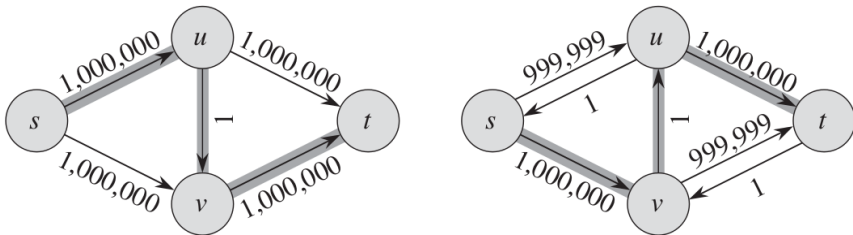
- Este grafo residual já não admite mais caminhos de aumento e o nosso método de Ford-Fulkerson fica por aqui!
- Este é o grafo residual do fluxo máximo que é de 23.

# Algoritmo de Ford-Fulkerson

- Seja qual for o caminho de aumento que formos escolhendo, é garantido que iremos convergir para o fluxo máximo (apenas no caso de os números serem irracionais é que isto não acontece, mas não iremos estudar casos destes em DAA).
- Para concretizar o método de Ford-Fulkerson precisamos de uma maneira de descobrir um caminho de aumento.
- Uma maneira possível é usar uma **pesquisa em profundidade**. Neste caso diz-se que estamos a usar o **algoritmo de Ford-Fulkerson**.
  - ▶ A complexidade de uma pesquisa em profundidade é de  $\mathcal{O}(|V| + |E|)$ , o que pode ser simplificado para  $\mathcal{O}(|E|)$  se admitirmos que o grafo é conexo (e nesse caso  $|E| \geq |V| - 1$ ).
  - ▶ Seja  $|f^*|$  o fluxo máximo. Apesar de um caminho de aumento fazer sempre o fluxo aumentar, pode aumentar muito "devagarinho". Se os números forem inteiros, pode aumentar apenas 1 de cada vez, pelo que a complexidade final do algoritmo é de  $\mathcal{O}(|E| \times |f^*|)$

# Algoritmo de Ford-Fulkerson

- A figura seguinte ilustra um grafo onde potencialmente o algoritmo de Ford-Fulkerson pode escolher caminhos que impliquem ir aumentando apenas 1 de cada vez, para um total potencial de 1 milhão de iterações antes de terminar



(imagem de Introduction to Algorithms, 3rd Edition)

- Precisamos de uma maneira melhor de procurar um caminho de aumento...

# Algoritmo de Edmonds-Karp

- Se para procurar um caminho de aumento usarmos uma **pesquisa em largura**, onde em cada passo procuramos o caminho mínimo (em termos de número de arestas), então estamos a usar o **algoritmo de Edmonds-Karp**.
  - ▶ Uma pesquisa em largura, tal como uma pesquisa em profundidade, demora  $\mathcal{O}(|E|)$
  - ▶ Usar sempre o caminho de aumento com menor número de arestas garante que apenas precisamos de o ir aumentando  $\mathcal{O}(|V| \times |E|)$  vezes (não iremos provar aqui isto, mas pode consultar o livro principal de DAA, o Introduction to Algorithms, para ver uma prova acessível).
  - ▶ Com isto, o algoritmo de Edmonds-Karp fica com uma complexidade total de  $\mathcal{O}(|V| \times |E|^2)$
- Existem ainda algoritmos de fluxo máximo mais rápido - ex: Dinic, que na sua versão "normal" demora  $\mathcal{O}(|E| \times |V|^2)$  - mas em DAA iremos ficar pelo Ford-Fulkerson e pelo Edmonds-Karp.

# Aplicações de Fluxo Máximo

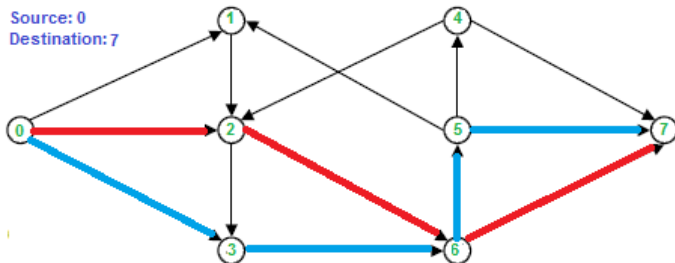
- Para terminar esta unidade, iremos falar de algumas aplicações possíveis de fluxo máximo, para além do caso onde estamos directamente a manipular uma rede de fluxo.
- Uma lista (não exaustiva) de aplicações:
  - ▶ Grafo com múltiplas origens e/ou múltiplos destinos
  - ▶ Encontrar número de caminhos que não usem as mesmas arestas
  - ▶ Encontrar o maior emparelhamento num grafo bipartido
  - ▶ ...



# Aplicações de Fluxo Máximo

## Encontrar número de caminhos que não usem as mesmas arestas

- Imagine que quer encontrar o máximo número de caminhos entre dois pontos tal que uma mesma aresta não possa aparecer em dois caminhos diferentes (*edge disjoint paths*). Basta para isso criar uma rede de fluxo onde todas as arestas têm capacidade 1 e desse modo o fluxo máximo dá-nos o número máximo de caminhos que não usem as mesmas arestas (pois a sua capacidade apenas "deixa passar" um caminho)



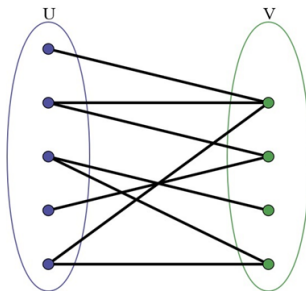
(imagem de geeksforgeeks)



# Aplicações de Fluxo Máximo

## Encontrar o maior emparelhamento num grafo bipartido

- Um **grafo bipartido** é um grafo onde os nós podem ser divididos em dois conjuntos de nós independentes, de tal maneira que uma qualquer aresta liga nós de conjuntos diferentes

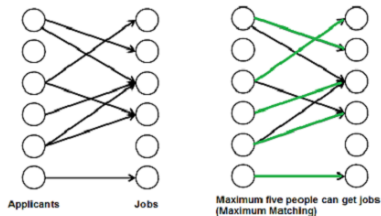


*(imagem de geeksforgeeks)*

# Aplicações de Fluxo Máximo

## Encontrar o maior emparelhamento num grafo bipartido

- Um **emparelhamento** num grafo bipartido é uma escolha de arestas tal que não exista mais do que uma aresta ligada a cada nó. Um **emparelhamento máximo** (*maximum bipartite matching*) é o emparelhamento de maior cardinalidade, ou seja, que use mais arestas.
- Imagine por exemplo uma situação onde temos candidatos e empregos, representados por um grafo bipartido, onde existe uma aresta a indicar se um dado emprego é ou não desejado por um dado candidato. Supondo que um emprego só pode dar para um candidato, e que um candidato só pode ficar com um emprego, um emparelhamento máximo dava-nos a maneira de alocar o maior número possível de empregos a candidatos.

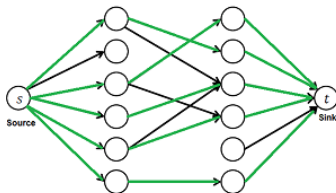


(imagem de geeksforgeeks)

# Aplicações de Fluxo Máximo

## Encontrar o maior emparelhamento num grafo bipartido

- Para descobrir um emparelhamento máximo, podemos simplesmente criar uma super origem ligada a todo o subconjunto de nós dos candidatos, e um super destino que recebe arestas de todos os empregos. Todas as arestas devem ter capacidade um e no final basta-nos calcular o fluxo máximo entre a super origem e o super destino!



The maximum flow from source to sink is five units. Therefore, maximum five people can get jobs.

(imagem de geeksforgeeks)